KHABEER (خبير): An Object-Oriented Arabic Expert System Shell

Mostafa M. Aref* and Husni A. Al-Muhtaseb

Information and Computer Science Department
King Fahd University of Petroleum and Minerals
P.O. Box 1658, Dhahran 31261
Saudi Arabia

e-mail: aref@ccse.kfupm.edu.sa
husni@ccse.kfupm.edu.sa

الخلاصة:

يعتبر نظام "خبير" أداة تعتمد على الذوات لبناء نظم الخبرة. ويوفر نظام خبير الأساسيات المطلوبة في نظم الخبرة وذلك كونه نظام إنتاج ويعتمد على الذوات ويوفر لغة استفسار متكاملة. كما يعتبر خبير أداة برجمة عربية حيث تتوفر التراكيب والأوامر وتنبيهات الأخطاء باللغة العربية. وقد تم تطوير خبير بلغة "سي" لتحقيق أهداف إمكانية النقل من حاسوب إلى آخر ورخص التكلفة وسهولة الاندماج مع نظم خارجية.

يستعمل خبير، كنظام إنتاج، طريقتين لتمثيل الحقائق: حقائق مرتبة (حقائق) و حقائق غبر-مرتبة (نماذج). وتعتبر الـ "قواعد" الطريقة الرئيسية لتمثيل المعرفة. يستعمل خبير الـ "محضر" لحفظ قائمة القواعد. وهناك سبع أساليب مختلفة لترتيب المحضر حيث يتم اختيار أول المحضر للتنفيذ.

تم تعريف 11 "صنف" في خبير للبرمجة بالذوات. ويمكن تعريف أصناف عقيمة واخرى منتجة ذات توارث متعدد. ولا يوجد حد لعدد السمات المعرفة في الصنف. ويتم إلحاق خصائص مختلفة بتعريف كل سمة. ومن هذه الخصائص: العدد والتخزين وتحصيل القيمة والتوارث. يمكن تعريف معالجات للأصناف المعرفة بأنواع أربعة. هناك العديد من الدوال للتعامل مع عينات الأصناف مثل عمل عينة وإعادة بدء عينة وقراءة وكتابة سمات وحذف عينة. تشمل لغة استفسار خبير على الكثير من الأسئلة الخاصة بعينات الأصناف مثل: هل-من-عينة و اوجد-عينة و اوجد-كل-العينات و نفذ-لعينة و نفذ-لكل-عينة و نفذ-لجميع-العينات.

ABSTRACT

KHABEER (خبير) is an object-oriented Arabic expert system shell. KHABEER provides the basic requirements of any expert system shell: production system, object-oriented and query language. KHABEER is an Arabic tool, where all the syntax, commands and error messages are in Arabic. KHABEER is written in C language to support the goals of high portability, low cost, and ease of integration with external systems.

KHABEER, as a production system, has two methods to represent facts: ordered facts (حقائق) and non-ordered facts (نماذج). Rules (قواعد) are the primary knowledge representation scheme in KHABEER. KHABEER uses agenda mechanism (محضر) for executing different rules. There are seven different strategies (اسلوب) for selection a rule to be fired.

KHABEER, as Object Oriented language, has 11 predefined classes and allows abstract and concrete class definitions and multiple inheritance. Only available memory limits the number of slots of an instance of a defined class in KHABEER. Various features of slots are supported by KHABEER. These features include default values, cardinality, storage, access, inheritance propagation and others. KHABEER allows the declaration of message-handlers for defined classes. Four types of message-handler declarations are allowed. Each type has its certain purpose. Manipulating instances of objects is supported through different functions in KHABEER. These functions include creating instances, re-initializing existing instances, reading slots, setting slots, deleting instances, instance query and other actions.

KHABEER, as a query language, provides six different types of queries. These queries, that concern instances (عينات) of classes, are: هل-من-عينة, اوجد-عينة, اوجد-كل-العينات, نفذ-لعينة, نفذ-لكل-عينة and نفذ-لجميع-العينات.

KHABEER (خبير): An Object-Oriented Arabic Expert System Shell

1. INTRODUCTION

The past decade has seen expert systems progress from effort in research laboratories to products built and deployed in industrial applications. Consequence to that, the number of tools for building expert systems has increased significantly. Many of these tools are written in languages other than LISP and executed on a variety of hardware platforms.

Expert systems tools are valuable because they provide rich software development environments, and the knowledge representation and the inference engine are already built into them [1-4]. KHABEER (خبير) is an Arabic CLIPS-based Expert System tool [5-8] where all the commands and syntax are written in Arabic. CLIPS (C Language Integrated Production System) is a C-based expert system tool developed by the Artificial Intelligence Section (now the software Technology Branch) at NASA's Johnson Space Center [9,10].

KHABEER was developed using the conventional language C. KHABEER uses rules as its primary knowledge representation approach and supports a rich pattern-matching language for specifying rule conditions. The system has interface that supports pull-down menus. In this paper, KHABEER as a production system is described in section 2 with the syntax and rules description. Section 3 presents the object oriented features of KHABEER version 2.0. The query language of the system is introduced in section 4. Section 5 presents several KHABEER examples and their outputs. Section 6 presents implementation issues and the integration of KHABEER with other programs. The conclusion and future work is given in section 7.

2. KHABEER AS A PRODUCTION SYSTEM

KHABEER may be considered as a production system [4], which provides pattern-directed control of a problem-solving process. KHABEER consists of knowledge base, fact list, agenda, and cycle of execution. The detail description of these components is as follows.

- The Knowledge base contains a set of production rules (قواعد). Each rule is a condition-action pair. The condition part of the rule is a pattern that determines when that rule may be applied. The action part defines the associated problem-solving step.

- The Fact list (the working memory) (الحقائق) contains a description of the current state of the problem. This description is a pattern that is matched against the condition part of the production rule. When the condition part of the production rule is matched by the contents of the working memory, the action part of that rule may be performed. The rule is said to be enabled (activated). Facts are the basic form of data in KHABEER. Each Fact is constructed of either several positional fields separated by spaces, or a word.

- The Agenda (محضر) is essentially a stack. Rules are pushed onto the stack when they are activated. If the priority of the new rule is less than the priority of the rule currently on the

top of the stack, the new rule is pushed down the stack until all rules of higher priorities are above it. Rules of equal or lower priorities remain below the new rule. The rules priorities can be assigned by the programmer.

- The Cycle of execution is the control structure of KHABEER. Once a knowledge base (production rules) is built and the fact list is prepared, KHABEER is ready to execute rules. The basic cycle of execution of KHABEER is as follows:

1- The knowledge base is examined to see if the conditions of any rule have been met.
2- All rules whose conditions, are currently met, are activated and placed on the agenda based on the conflict resolution strategy.
3- The top rule on the agenda is selected, , and its actions are executed.

As a result of actions execution, new rules can be activated or deactivated. This cycle is repeated until all rules that can be fired have done so or the rule limit is reached. The number of rule firings allowed in a cycle may be set by the programmer.

2.1 KHABEER SYNTAX

KHABEER has a Lisp-like syntax as shown in Figure 1. It supports a rich pattern-matching language for specifying rule conditions. The pattern-matching language operates on both single fields (expressed as ؟ or ؟متغير) and multifield (expressed as ؟# or #؟متغير-متعدد) sequences composed of strings, symbols and numbers. KHABEER pattern-matching operators range from a single operator that will match any and every fact in the knowledge base to operators that only match facts that meet specific constraints. Conditions can also be written such that a rule is activated only if a pattern cannot be matched by any fact in the knowledge base. Thus, reasoning can be based on the absence of information as well as its presence.

KHABEER also supports templates (نماذج) as a means of specifying rule conditions. Templates, frame-like structures composed of named slots with values, support the specification of default values and metaknowledge in the form of type information.

The condition side of KHABEER rules has an implicit logical AND between conditions (ظروف). KHABEER also supports the specification of explicit logical AND (و) and OR (او) conditions for the condition-side of rules. If the conditions are specified as disjunctions (using an explicit OR (او)), the rule is a candidate to fire if any of the disjuncts are matched by facts.

```
(عرف-قاعدة اسم-القاعدة                    "معلومات اضافية للتعريف بالقاعدة"
                                          ؛ الجانب الاول من القاعدة
(ظرف-1)
                                          ؛ يحتوي على عدة ظروف
(ظرف-2)
                                          ؛ كل ظرف بين قوسين
(ظرف-3)
```

```
=<
        (فعل-1)                    ؛ الجانب الاخر من القاعدة
        (فعل-2)                    ؛ يحتوي على عدة افعال
        (          (فعل-3)
```

Figure 1. Syntax of عرف-قاعدة Construct

In addition, KHABEER provides procedural programming constructs (if .. then .. else) (اذا،،فان،،والا), while (طالما) on the action side of the rules. KHABEER provides debugging aids which include commands that produce a trace of facts asserted in the knowledge base (راقب حقائق), rules placed on the agenda (راقب قواعد), and rules that fire (راقب تنفيذ). Break points (ضع-وقفة) can be specified contingent on specific rules firing. A number of commands are available for displaying entities in the knowledge base such as:

(حقائق) displays the facts,
(قواعد) displays the rules in the knowledge base,
(محضر) displays the rules in the agenda,
(طابق) displays a list of facts that match each condition of a specified rule.

The (شغل) command can be executed with a positive integer that specifies the number of rules to be fired, (شغل 1) results in single-step execution. KHABEER provides seven conflict resolution strategies to put the activated rules of equal priority (اولوية). These strategies are as follows:

- **اسلوب عميق** (depth): Newly activated rules are placed above all rules of the same priority.
- **اسلوب سطحي** (breadth): Newly activated rules are placed below all rules of the same priority.
- **اسلوب تبسيطي** (simplicity): Newly activated rules are placed above all activations of rules with equal or higher specificity (تخصيص). The specificity of a rule is determined by the number of comparisons that must be performed on the first side of the rule.
- **اسلوب تركيبي** (complexity): Newly activated rules are placed above all activations of rules with equal or lower specificity.
- **اسلوب احدث-اخص** (LEX): Newly activated rules are placed using the OPS5 strategy LEX. First the recency of fact indices is used to determine where to place the activation. An activation with a more recent fact index is placed before activations with less recent fact indices (احدث). If two activations have the exact same recency, the activation with the higher specificity is placed above the activation with the lower specificity (اخص).
- **اسلوب احدث-ظرف1** (MEA): Newly activated rules are placed using the OPS5 strategy MEA. First the recency of the fact index associated with the first pattern is used to determine where to place the activation. If two activations have the same fact index for the first pattern, then the LEX strategy is used to determine placement of the activation.

- **اسلوب عشوائي** (random): Each activation is assigned a random number which is used to determine its placement among activations of equal priority.

The default strategy is depth. The current strategy can be set by using the **اسلوب** command (which will reorder the agenda based upon the new strategy).

2.2 KHABEER TERMS & VOCABULARY المصطلحات في خبير

KHABEER vocabulary and terms were chosen using the suggested headline points in [6]. Some of these points are:

- Use the smallest possible number of words such that the meaning will not be misunderstood.
- Delete some functions that are not related to Arabic language such as *uppercase* and *lowercase*.
- Use the shortest of the imperative (امر) form and the gerund (مصدر) form. If their lengths are equal, use the one which starts with uncommon letter.
- Give the terms their actual and practical meanings which may be different than the "dictionary" meaning.
- Do not use abbreviations.
- Better to translate a negative word into a single Arabic word. For example *unusual* is translated to "شاذ " and not to "غير عادي".
- Some terms needs to be replaced totally. Left parenthesis is given the term قوس الافتتاح.

For more comprehensive details the reader may refer to [6].

3. KHABEER OBJECT ORIENTED LANGUAGE برمجة الذوات في خبير

KHABEER supports Object oriented Language features. The primary five characteristics of any object oriented language are [10]:

- **abstraction (تجرد)**: is a higher level, more intuitive representation for a complex concept;
- **encapsulation (تغليف)**: is the process whereby the implementation details of an object are masked by a well-defined external interface;
- **inheritance (وراثة)**: where classes may be described in terms of other classes by use of inheritance;
- **polymorphism (تعدد التصرف)**: is the ability of different objects to respond to the same message in a specialized manner; and
- **dynamic binding (ربط متغير)**: is the ability to defer the selection of which specific message-handlers will be called for a message until run-time.

In KHABEER, the definitions of new classes (اصناف) allows the abstraction of new data types. The slots (سمات) and message-handlers (معالجات) of these classes describe the properties and behavior of a new group of objects. KHABEER supports encapsulation by requiring message-passing for the manipulation of instances of user-defined classes. An instance (عينة) cannot respond to a message for which it does not have a defined message-handler.

The user is allowed to specify some or all of the properties and behavior of a class in terms of one or more superclasses (فصائل). This process is called multiple inheritance (متعدد الارث). KHABEER uses the existing hierarchy of classes to establish a linear ordering called the class precedence list (قائمة الترتيب الوراثي) for a new class. Objects which are instances of this new class can inherit properties (سمات) and behavior (معالجات) from each of the classes in the class precedence list. The word precedence implies that properties and behavior of a class first in the list override conflicting definitions of a class later in the list.

Polymorphism implies that one KHABEER object can respond to a message in a completely different way than another object. This is accomplished by attaching message-handlers with differing actions but which have the same name to the classes of these two objects respectively. An object reference in ارسل (send) function call is not bound until run-time. This is called dynamic binding. For example, an instance name or variable might refer to one object at the time a message is sent and another at a later time.

A query system for determining, grouping and performing actions on sets of instances of user-defined classes that meet user-defined criteria is provided by KHABEER. The query system allows the user to associate instances that are either related or not. The user can use the query system to determine if a particular association set exists, he can save the set for future reference, or he can iterate an action over the set. KHABEER query language is discussed in section 4.0.

3.1 PREDEFINED SYSTEM CLASSES اصناف النظام

KHABEER provides eleven system classes (shown in Figure 2): Object (ذات_), User (مستخدم_), Primitive (اولي_), External-Address (عنوان-خارجي_), Multifield (حقول_), Number (رقم_), Integer (صحيح_), Float (حقيقي_), Lexeme (مفردة_), Symbol (رمز_) and String (سلسلة_). These classes are abstract classes. Thus, they are used only for inheritance. The Object (ذات_) class is a superclass of all other classes including user-defined classes. A predefined class can not be modified nor deleted by a user.

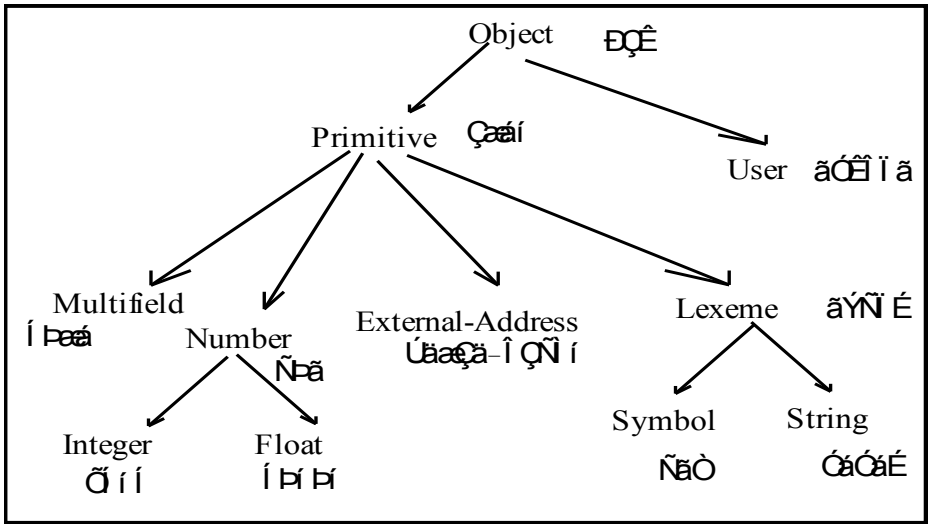

Figure 2. Relationships between Classes

3.2 DEFINING CLASSES تعريف الاصناف

Classes can be defined using عرف-صنف (define class) construct (as shown in Figure 3). This construct consists of four elements: a name, a list of superclasses for which the new class inherits slots and message-handlers, a specifier defining whether or not the creation of direct instances of the new class is allowed, and a list of slots specific to the new class.

```
(يكون <اسم فصيلة> +)][<ملحوظة>[ (عرف-صنف <اسم الصنف>
                                    <تحديد السمة>*)] <تحديد وظيفة> [
<تحديد وظيفة>  (منتج) | (عقيم) =::
  <تحديد السمة>  (<خصائص السمة>  <اسم السمة>  (سمة =::
]<وراثة>[ | ]<مسلك>[ | ]<تخزين>[ | ]<عدد>[ | ]<مفترض>[<خصائص السمة> =::
                                    | ]<مصدر>[
    <مفترض>  (مفترض-متغير <معادلة>*) | (مفترض <معادلة>*)  =::
      <عدد>  (مفرد) | (متعدد)  =::
    <تخزين>  (محلي) | (مشترك)  =::
    <مسلك>  (تحضر-فقط) | (تقرأ-تكتب) | (تقرأ-فقط)  =::
    <وراثة>  (تورث) | (لاتورث)  =::
    <مصدر>  (مقيد) | (مركب)  =::
```

The underlined values are the default values.

Figure 3. Syntax of عرف-صنف Construct

Redefining an existing class deletes the current subclasses and all associated message-handlers. An error will occur if instances of the class or any of its subclasses exist. Any old message-handlers for the class which do not conflict with implicit slot-accessor message-handlers in the new definition are reattached.

3.2.1 Multiple Inheritance تعدد الارث

صنف (class) in KHABEER inherits from فصيلة (superclass). Every user-defined class must have at least one direct superclass. When a class has more than one direct superclass multiple inheritance occurs. KHABEER establishes قائمة الترتيب الوراثي (a class precedence list) by examining the direct superclass list for a new class. The new class inherits slots and message-handlers from each of the classes in the class precedence list. Slots and message-handles of a class in the list override conflicting definitions of another class found later in the list. A specific (محدد) class is a class that comes before another class in the list. وصف-صنف function can be used to list the class precedence list.

3.2.2 Abstract and Concrete Classes الاصناف العقيمة والمنتجة

No direct instances of a class can be created if this class is عقيم (abstract). A class of type منتج (concrete) can have direct instances. By default, a new class is منتج.

3.2.3 Slots and their facets السمات و خصائصها

Values associated with instances (عينات) of a user-defined class are stored in slots (سمات). To determine the set of slots for an instance, the class precedence list for the instances is examined in order from

most specific to most general (right to left). A class is more specific than its superclasses. Slots specified in any of the classes in the class precedence list are given to the instance, with the exception of no-inherit (لاتورث) slots. If a slot is inherited from more than one class, the definition given by the more specific class takes precedence, with the exception of composite (مركب) slots.

Facets (الخصائص) describe various features of a slot. These facets are: مفترض (default value), عدد (cardinality), تخزين (storage), مسلك (access), وراثة (inheritance propagation) and مصدر (source) of other facets. With the exception of shared slots (سمات مشتركة), each object can still have its own value for a slot .

Default Value Facets خصائص القيم المفترضة

The facets مفترض (default) and مفترض-متغير (default-dynamic) can be used to specify an initial value given to a slot when an instance is created or initialized. The specified expression in مفترض-متغير is evaluated every time an instance is created, and the result is assigned to the appropriate slot.

Cardinality Facets خصائص العدد

The facet متعدد (multiple) specifies that a slot can hold zero or more values, and the facet فرد (single) specifies that the slot can hold zero or one value. Slots with متعدد facets are called حقول slots. حقول slot values can be manipulated with the standard حقول functions, such as عنصر and طول. KHABEER also provides functions for setting حقول slots.

Storage Facets خصائص التخزين

The facet محلي (local) specifies that the value be stored with the instance. The facet مشترك (shared) specifies that the value be stored with the class. In the محلي facet, each instance can have a separate value for the slot. in the other facet, all instances will have the same value for the slot.

Access Facets خصائص المسلك

The access facets types are تقرا-تكتب (read-write), تقرا-فقط (read-only), and تحضر-فقط (initialize-only) where the slot can be read and set by slot overrides in عمل-عينة call and جهز message-handlers.

Inheritance Propagation Facets خصائص الوراثة

The facet تورث (inherit) specifies whether a slot in a class can be given to instances of other classes that inherit from the first class or not. The facet لاتورث (no-inherit) says that only direct instances of this class will get the slot.

Source Facets خصائص المصدر

The مقيد (exclusive) facet says that take the facets from the most specific class which gives the slot and give default values to any unspecified facets. The مركب (composite) facet causes facets which are not explicitly specified by the most specific class to be taken from the next most specific class.

3.3 DEFINING MESSAGE-HANDLERS تعريف المعالجات

The construct عرف-معالج is used for specifying the behavior of a class of objects in response to a particular message (shown in Figure 4). This construct consists of the following seven elements: a class name (اسم الصنف) to which attach the handler , a message name (اسم الرسالة) to which the handler will response, a handler type, an optional comment, a list of parameter that will be passed to the handler during execution, an optional wild card parameter, and series of expressions which are executed when the handler is called.

```
(عرف-معالج <اسم الصنف> [<نوع المعالج>] [<ملحوظة>]
   <اسم الرسالة>
   ( <فعل>* ) [متغير-عام <متغير>] (<متغير>*)
<متغير-احادي> ::= <متغير>
<نوع-المعالج> ::= قبل | بعد | رئيسي | حول | خلال
<متغير-حقول> ::= <متغير-عام>
```

Figure 4. Syntax of Message handlers.

Message handlers are uniquely identified by class, name and type. All message handlers have an implicit parameter called نفس؟ (self) which binds the active instance for a message. This parameter name is reserved and cannot be explicitly listed in the message handler's parameter. There are three primary message handlers that are attached to the class مستخدم_: تحضير (initialize), حذف (delete), and طبع (print).

There are four categories of message handlers: رئيسي (primary), قبل (before), بعد (after) and حول (around). The return values of قبل and بعد handlers are always ignored. قبل handlers execute before the رئيسي ones, and بعد message-handlers execute after the رئيسي ones. The return value of a message is generally given by the رئيسي message-handlers, but حول handlers can also return a value. حول message-handlers allow the user to wrap code around the rest of the handlers. They begin execution before the other handlers and pick up again after all the other message-handlers have finished.

The body of عرف-معالج is a sequence of expressions that are executed in order when the handler is called. عرف-معالج returns the value of the last expression in the body. The body of عرف-معالج may directly manipulate slots of the active instance.

3.3.1 Slot accessor Handlers معالجة السمات

For every slot in عرف-صنف two primary message-handlers are created implicitly: حصل-<اسم سمة> to read slot values in instances of a class and ضع-<اسم سمة> to set slot values in instances of a class. حصل-<اسم سمة> handler returns the value of the slot, or the symbol خطأ if the slot has no value. ضع-<اسم سمة> returns the symbol صح if the slot was successfully set, or the symbol خطأ otherwise.

3.3.2 Predefined System Message handlers المعالجات المعرفة

KHABEER has three primary message-handlers that are attached to the class مستخدم_. These handlers are جهز, حذف, and طبع. These handlers cannot be deleted or modified. The first handler جهز is used for instance initialization with class default values after creation. The second handler حذف is used for instance deletion. The third handler طبع is used for displaying slots of an instance and their values.

3.4 MESSAGE DISPATCH انجاز الرسالة

KHABEER uses the roles (حول (around), قبل (before), رئيسي (primary) and بعد (after)) to establish a complete set of message handlers which are applicable to a given message (sent by the command ارسل). This is done by examining قائمة الترتيب الوراثي (class precedence list) of the active instances class. This process is referred to as the message dispatch (انجاز الرسالة).

3.4.1 Message handler Precedence اولويات المعالجات

The order of execution of message handlers begins with حول handlers from most specific to most general, then قبل handlers execute from most specific to most general, then رئيسي handlers begin execution from most specific to most general, after they finish execution from most general to most specific, بعد handlers execute from most general to most specific and حول handlers finish execution from most general to most specific.

3.5 MANIPULATING INSTANCES معالجة العينات

Manipulation of objects is done by sending them messages. This is achieved by using ارسل (send) function (shown in Figure 5), which takes as arguments the destination object for the message, الرسالة (message) itself and any arguments which are to be passed to handlers. The return value of ارسل is the result of the message.

```
(ارسل  <تعبير-صنف>  <تعبير-اسم-الرسالة>  <تعبير> *)
```

Figure 5. Syntax of ارسل

The slots of ذات (object) may be read or set directly only within the body of a message handler that is executing on behalf of a message that was sent to that object. By this way, KHABEER implements the notion of encapsulation. Any action performed on an object by an external source must be done with messages. Creation and initialization of an instance of a user defined class are performed by the function عمل-عينة (make instance).

3.5.1 Creating Instances عمل عينات

Instances (عينات) of user defined classes (اصناف) must be explicitly defined by the user. All instances are deleted during حضر (reset) command, and they can be loaded and saved similarly to facts. All operations involving instances require message passing using ارسل (send) function except for creation.

```
( <تعريف-عينة>  (عمل-عينة
<تعريف-عينة> ::= <تعبير-اسم-عينة> من <تعبير-اسم-صنف>  <اجتياز-سمة>*
( *<تعبير>  <تعبير-اسم-عينة>) ::= <اجتياز-سمة>
```

Figure 6. Syntax of عمل-عينة

A function called عمل-عينة (make instance) is used to create and initialize a new instance (shown in Figure 6). This function sends an initialization message to the new object after allocation, and the user can customize instance initialization. عمل-عينة allows changing any predefined initialization for a particular instance.

عمل-عينة returns the name of the new instance on success or the symbol خطأ (false) on failure. The evaluation of <تعبير-اسم-عينة> can either be an instance name or a symbol. When عمل-عينة creates a new instances it performs the following steps:

1) If the instance exists, that instance receives a delete message, (ارسل <اسم-عينة> حذف).
2) An uninitialized instance of the specified class is created.
3) All اجتياز-سمة (slot overrides) are evaluated and placed by -ضع messages, e.g.
   (ارسل <اسم-عينة> ضع-<اسم-سمة><تعبير>*).
4) The new instance receives the جهز message, e.g. (ارسل <اسم-عينة> جهز) for initialization. The handler attached to class مستخدم_ will respond to this message. This handler calls the -جهز سمات function. This function uses defaults from the class definition - if any - for any slots which do not have اجتياز-سمة . The class defaults are placed directly without the use of messages.

Defining instances Construct تعريف العينات

عرف-عينات construct allows the specification of instances which will be created every time حضر (reset) command is executed (shown in Figure 7). Whenever حضر is issued, all current instances receive حذف message, and the equivalent of عمل-عينة function call is made for every instance specified in عرف-عينات (define instances) constructs. Instances of عرف-عينات are created in order, and if any individual creation fails, the remainder of the instances will be aborted.

```
(عرف-عينات <اسم-تعريف-عينات> [<ملحوظة> <نموذج-عينة>*]
  <نموذج-عينة> ::= (<تعريف-عينة>)
```

Figure 7. Syntax of عمل-عينات

3.5.2 Re-initializing Existing Instances تجهيز العينات

To provide the ability to reinitialize an existing instance with class defaults and new slot overrides, the function جهز-عينة is used (shown in Figure 8). The return value of جهز-عينة is the name of the new instance on success or the symbol خطأ (false) on failure. The evaluation of <تعبير-اسم-عينة> can either be an instance name or a symbol.

```
(جهز-عينة <تعبير-اسم-عينة> <اجتاز-سمة> *)
```

Figure 8. Syntax of جهز-عينة

3.5.3 Reading and setting Slots قراءة وتعيين السمات

Rules, defined functions or any sources external to an object, can read or write an objects slots only by sending the object حصل- (get) or ضع- (put) messages. Message handles executing on the behalf of an object can either use messages or direct access to read the objects slots. An attempt to read a slot which does not have a value will generate an error. There are ways of testing the existence of slots and their values.

3.5.4 Deleting Instances حذف العينات

Sending حذف (delete) message to an instance removes it from the system (shown in Figure 9). Within a message handler, حذف-عينة function can be used to delete the active instance for a message.

```
(حذف-عينة)
(ارسل  <عينة>  حذف)
```

Figure 9. Syntax of حذف-عينة

4.0 KHABEER QUERY LANGUAGE

KHABEER has a useful query system for determining and performing actions on sets of instances of user defined classes. The instance query system in KHABEER provides six functions. These six functions are as follows.

Function	**Purpose**
هل-من-عينة	finds if one or more instance sets satisfy a query.
اوجد-عينة	gives the first instance set that satisfies a query.
اوجد-كل-العينات	Groups and returns all instance sets which satisfy a query.
نفذ-لعينة	Performs an action for the first instance set which satisfies a query.
نفذ-لكل-عينة	Performs an action for every instance set which satisfies a query as they are found.
نفذ-لجميع-العينات	Groups all instance sets which satisfy a query and then iterates an action over this group.

The syntax of the query includes the name of the query, instance set constrains, query conditions and query actions (shown in Figure 10). فئة العينات (instance set) is an ordered collection of instances of a set of classes defined by the user. KHABEER uses straightforward permutations to generate instance sets. ظروف الاستفسار (queries) are user defined Boolean expressions applied to an instance set to determine if the instance set meets further user defined restrictions. If the evaluation of these expressions for an instance set is anything but the symbol خطا (false), the instance set is said to satisfy the query. Since only instance sets which satisfy a query are of interest, and the query is evaluated for all possible instance sets, the query should not have any side effects.

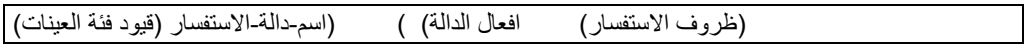

Figure 10. Query Syntax

افعال الدالة (distributed actions) are a user-defined expressions evaluated for each instance set which satisfies a query. Unlike queries, distributed actions must use messages to read slots of instance set members. If more than one action is required. An instance set query function can be called from anywhere that a regular function can be called. If a variable from an outer scope is not masked by an instance set member variable, then that variable may be referenced within the query and action. In addition, rebinding variables within an instance set function action is allowed. However, attempts to rebind instance set member variables will generate errors. Binding variables are not allowed within a query. Instance set query functions can be nested.

Instance set member variables are only in scope within the instance set query function. Attempting to use instance set member variables in an outer scope will generate an error. If an error occurs during an instance set query function, the function will be immediately terminated and the return value will be the symbol خطا (false). The instance query system in KHABEER provides six functions. For a given set of instances, all six query functions will iterate over these instances in the same order. However, if a particular instance is deleted and recreated, the iteration order will change.

4.1 هل-من-عينة: This function applies a query to each instance set which matches the template (shown in Figure 11). If an instance set satisfies the query, then the function is immediately terminated, and the return value is the symbol صح (true). Otherwise, the return value is the symbol خطا (false).

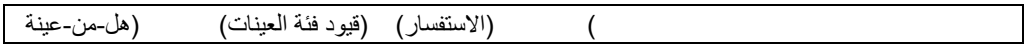

Figure 11. Syntax of هل-من-عينة

4.2 اوجد-عينة: This function applies a query to each instance set which matches the template (shown in Figure 12). If an instance set satisfies the query, then the function is immediately terminated, and the instance set is returned in a multifield value. Otherwise, the return value is a zero-length multifield value. Each field of the multifield value is an instance name representing an instance set member.

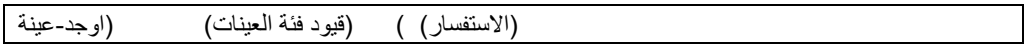

Figure 12. Syntax of اوجد-عينة

4.3 اوجد-كل-العينات: This function applies a query to each instance set which matches the template (shown in Figure 13). Each instance set which satisfies the query is stored in a multifield value. This multifield value is returned when the query has been applied to all possible instance sets. If there are n instances in each instance set, and m instance sets satisfied the query, then the length of the returned multifield value will be n * m. The first n fields correspond to the first instance set, and so on. Each field of the multifield value is an instance-name representing an instance set member. The

multifield value can consume a large amount of memory due to permutational explosion, so this function should be used judiciously.

```
(الاستفسار )  (اوجد-كل-العينات  (قيود فئة العينات)
```

Figure 13. Syntax of اوجد-كل-العينات

4.4 نفذ-لعينة: This function applies a query to each instance set which matches the template (shown in Figure 14). If an instance set satisfies the query, the specified action is executed, and the function is immediately terminated. The return value is the evaluation of the action. If no instance set satisfied the query, then the return value is the symbol خطا (false).

```
(نفذ-لعينة  (  (افعال)  (الاستفسار)  (قيود فئة العينات)
```

Figure 14. Syntax of نفذ-لعينة

4.5 نفذ-لكل-عينة: This function applies a query to each instance set which matches the template (shown in Figure 15). If an instance set satisfies the query, the specified action is executed. The return value is the evaluation of the action for the last instance set which satisfied the query. If no instance set satisfied the query, then the return value is the symbol خطا (false).

```
(الاستفسار)  (افعال)  )  (قيود فئة العينات)  (نفذ-لكل-عينة
```

Figure 15. Syntax of نفذ-لكل-عينة

4.6 نفذ-لجميع-العينات: This function is similar to نفذ-لكل-عينة except that it groups all instance sets which satisfy the query into an intermediary multifield value (shown in Figure 16). If there are no instance sets which satisfy the query, then the function returns the symbol خطا (false). Otherwise, the specified action is executed for each instance set in the multifield value, and the return value is the evaluation of the action for the last instance set to satisfy the query. The intermediary multifield value is discarded. This function can consume large amounts of memory in the same fashion as اوجد-كل-العينات. This function should be used in lieu of نفذ-لكل-عينة when the action applied to one instance set would change the result of the query for another instance set (unless that is the desired effect).

```
(نفذ-لجميع-العينات  (قيود فئة العينات)  (الاستفسار)  (افعال)  )
```

Figure 16. Syntax of نفذ-لجميع-العينات

5.0 EXAMPLES

To show the syntax of KHABEER, several examples are given bellow. The output of each example is shown after it.

5.1 A Simple Example

This example shows a simple rule and two facts. The rule will be activated and fired. A new fact is added to the fact list (shown in Figure 17).

```
(عرف-حقائق حالة-الثلاجة (انوار الثلاجة مضاءة)
                         (باب الثلاجة مفتوح))
مثال-قاعدة          (عرف-قاعدة
          (انوار الثلاجة مضاءة)
          (باب الثلاجة مفتوح)
<=        (ضف (اكل الثلاجة تالف)))
```

```
خبير> (حضر)
خبير> (نفذ)
خبير> (حقائق)
(ح-0 (حقيقة-اولية))
(ح-1 (انوار الثلاجة مضاءة))
(ح-2 (باب الثلاجة مفتوح))
(ح-3 (اكل الثلاجة تالف))
مجموع-الحقائق 4
```

Figure 17. Simple example

5.2 An Example on Rules definitions

This example shows a set of 6 rules. These rules are applied on different set of facts and the response of these rules are shown as output. The rules concern the diagnosis of a pump (shown in Figure 18).

```
))الحالة حقل(  المضخة          عرف-نموذج(
عيب-في-النظام-1          عرف-قاعدة(
          )غير-معروف حالة-الخطا(
          )مكسور الصمام(  )عالية درجة-الحرارة(  او(
)))مغلقة الحالة(المضخة (
<=        ))سطر "النظام يحتوي علي عيب1" ش طبع(

عيب-في-النظام-2          عرف-قاعدة(
          )غير-معروف حالة-الخطا(
          ))مغلقة الحالة(المضخة (
<=        ))سطر "النظام يحتوي علي عيب2" ش طبع(

عيب-في-النظام-3          عرف-قاعدة(
```

```
          )غير-معروف حالة-الخطا(
          )مكسور الصمام(
<=        ))سطر "النظام يحتوي علي عيب3" ش طبع(

عيب-في-النظام-4          عرف-قاعدة(
          )غير-معروف حالة-الخطا(
          )عالية درجة-الحرارة(
<=        ))سطر "النظام يحتوي علي عيب4" ش طبع(

عيب-في-النظام-5          عرف-قاعدة(
          )مثبتة حالة-الخطا(
          ))مغلق الصمام(          )عالية درجة-الحرارة(          و(          او(
          )))مفتوح الصمام(          )منخفضة درجة-الحرارة(          و(
<=        ))سطر "النظام يحتوي علي مشكلة" ش طبع(

المعدل-العالى          عرف-قاعدة(
          )عالية درجة-الحرارة(
          )مفتوح الصمام(
          ))مثبتة حالة-الخطا(ليس(
<=        ))سطر "ينصح باغلاق الصمام لان درجة الحرارة عالية" ش طبع(
)غير-معروف عرف-حقائق حالة-المضخة (حالة-الخطا(
                              )عالية درجة-الحرارة (
                              ))مفتوحة الحالة (المضخة (
                              ))مغلق الصمام (
```

```
خبير> (حضر)
خبير> (نفذ)
النظام يحتوي على عيب1
النظام يحتوي على عيب4
```

Figure 18. Example of rules definition

```
)مثبتة  حالة-الخطا(          (عرف-حقائق حالة-المضخة
                              )عالية درجة-الحرارة (
                              ))مفتوحة الحالة (المضخة (
                              ))مغلق الصمام (
```

```
خبير> (حضر)
خبير> (نفذ)
النظام يحتوي على مشكلة
```

```
)غير-مثبتة (عرف-حقائق حالة-المضخة (حالة-الخطا
                              )عالية درجة-الحرارة (
                              ))مفتوحة الحالة (المضخة (
                              ))الصمام مفتوح (
```

```
خبير> (حضر)
خبير> (نفذ)
ينصح باغلاق الصمام لان درجة الحرارة عالية
```

Figure 18. Example of rules definition (continuation)

5.3 An Example on instances and message handlers

This example shows the definition of a class and a message handler. Then, an instance is made of that class. The contents of the instance is filled by the message handler and is printed out (shown in Figure 19).

```
(عرف-صنف (سيارة يكون مستخدم)
        (سمة عرف-المقعد-الامامي)
        (سمة صندوق متعدد))
        (سمة عدد-الادوات-في-الصندوق))
(معالج عرف سيارة سيارة ضع-المواد-في-السيارة #مادة؟ البقية؟)
                (ارسل نفس ضع-المقعد-الامامي مادة؟)
                (ارسل نفس ضع-صندوق البقية؟)
                (ارسل نفس ضع-عدد-الادوات-في-الصندوق طول البقية؟)))
```

```
(خبير> سيارة من تويوتا عمل-عينة)
[ تويوتا]
(خبير> ارسل [تويوتا] ضع-المواد-في-السيارة شنطة اطار شنطة-دبلوماسية)
نعم
(ارسل خبير> [تويوتا] طبع)
سيارة من تويوتا
( المقعد-الامامي شنطة)
(صندوق اطار شنطة-دبلوماسية)
(عدد-الادوات-في-الصندوق 2)
```

Figure 19. Example on instances and message handlers

```
(عرف-صنف شخص يكون (عقيم مستخدم) (تقرا-فقط جنس سمة) (مشترك ) (عمر سمة))
(عرف-صنف انثى يكون (عقيم شخص) ( جنس سمة مركب) (انثى مفترض)))
(عرف-صنف ذكر يكون (شخص عقيم) (جنس سمة مركب) (ذكر مفترض )))
(عرف-صنف بنت يكون (انثى) (عمر سمة مفترض 4) (رقم نوع -) (مدى 0.0 17.9)))
(عرف-صنف إمرأة يكون (انثى) (عمر سمة مفترض 25) (رقم نوع -) (مدى 18.0 100.0)))
(عرف-صنف ولد يكون (ذكر) (عمر سمة مفترض 4) (_رقم نوع) (مدى 0.0 17.6)))
(عرف-صنف رجل يكون ذكر) (عمر سمة مفترض 25) (رقم نوع -) (مدى 18.0 100.0)))
(عرف-عينات اشخاص
        (من رجل-1 رجل 18 عمر)) (من رجل-2 رجل 60 عمر))
        (من إمرأة-1 إمرأة 18 عمر)) (من إمرأة-2 إمرأة 60 عمر))
        (من إمرأة-3 إمرأة)
        (ولد-1 من ولد-2) (ولد 8 عمر)) (ولد من ولد-4)
        (ولد-3 من ولد)
        (بنت-1 من بنت-2) (بنت 8 عمر))
```

```
(حضر خبير>)
(نفذ-لكل-عينة خبير> (رجل ولد ؟رجل-او-ولد) (إمرأة بنت ؟إمرأة-او-بنت))
        (= ارسل (رجل-او-ولد ؟حصل-عمر) (ارسل إمرأة-او-بنت ؟حصل-عمر))
        (ش طبع "(" ولد-او-رجل؟ "،" بنت-او-إمرأة؟ " ) " سطر))
([بنت-1]،[ولد-1])
```

```
([ولد-2]،[بنت-2])
([ولد-3]،[بنت-2])
([ولد-4]،[بنت-2])
([رجل-1]،[إمرأة-1])
([رجل-1]،[إمرأة-2])
))إمرأة بنت ؟إمرأة-او-بنت) (رجل ولد ؟رجل-او-ولد(( نفذ-لكل-عينة(خبير>
        )؟إمرأة-او-بنت:عمر (= ؟رجل-او-ولد:عمر
        ))سطر  ")"  ؟إمرأة-او-بنت  "،"  ؟رجل-او-ولد "( " ش طبع(
([ولد-1]،[بنت-1])
([ولد-2]،[بنت-2])
([ولد-3]،[بنت-2])
([ولد-4]،[بنت-2])
([رجل-1]،[إمرأة-1])
([رجل-1]،[إمرأة-2])
```

```
)؟صنف  (حساب-العينات  عرف-دالة(
        ؟حساب  0  ) قيد(
        )؟حساب قيد(  نعم  ))  ؟صنف ؟عينة((  نفذ-لكل-عينة(  1  ؟حساب +  (((
        (؟حساب
)؟صنف(حساب-العينات-2  عرف-دالة(
        )طول(  ؟صنف ؟عينة((اوجد-كل-العينات  ؟(((نعم))
```

Figure 20. Example on Query Language

```
)إمرأة  حساب-العينات(خبير>
3
)ولد  حساب-العينات(خبير>
4
)حساب-العينات-2  إمرأة(خبير>
3
)حساب-العينات-2  ولد(خبير>
4
)) 30 ؟ر:عمر  رجل)) (> ؟ر  ((هل-من-عينة(خبير>
نعم
))؟م:عمر  ؟ر:عمر (=  ))  إمرأة ؟م)  (رجل ؟ر((اوجد-عينة  (خبير>
[إمرأة-1]) ([رجل-1]
))؟م:عمر  ؟ر:عمر  (=  ))إمرأة ؟م)  (رجل ؟ر((اوجد-كل-العينات (خبير>
[إمرأة-1]) [إمرأة-1] [رجل-1] ([رجل-1]
))شخص ؟ش3) (شخص ؟ش2) (شخص ؟ش1((نفذ-لعينة  (خبير>
        )؟ش3:عمر ؟ش2:عمر ؟ش1:عمر = (و(
            ؟ش1  ؟ش2) خلاف(
            ؟ش1  ؟ش3) خلاف(
            ؟ش2  ؟ش3) خلاف(
        ))؟ش3  سطر  ؟ش2  " و "  ؟ش1  " و " ش طبع(
[ولد-2] و[ولد-3] و [بنت-2]
خبير> (عينات)
صنف-بنت:
بنت-1
بنت-2
```

```
صنف-ولد
ولد-1
ولد-2
ولد-3
ولد-4
صنف-إمرأة:
إمرأة-1
إمرأة-2
إمرأة-3
صنف-رجل:
رجل-1
رجل-2
مجموع-العينات 11
```

Figure 20. Example on Query Language (continuation)

5.4 An Example on Query Language

This example shows the definition of several classes and several instances of these classes. KHABEER query language is used to request some information. Some functions are defined and applied on these instances. At the end, the command (عينات) is used to print all instances in KHABEER (shown in Figure 20).

6.0 IMPLEMENTATION ISSUES

KHABEER consists of 63 C files and 62 header files. The size of all source code files is 2.3 Mbytes. Microsoft C++ is used to compile KHABEER source files. The size of the executable code of KHABEER which support object oriented language is 1.3M bytes. The system runs under MS-Windows 3.1. Error messages within KHABEER system are generated in Arabic. Reference Manual for the system is under development [7]. There are three ways for integrating KHABEER with other programs. First, since KHABEER is written in C, an integrating subprogram may be written in C and complied with the system. Second, a Dynamic Linked Library (DLL) may be generated for KHABEER. Therefore other external system may use it. Third, since KHABEER is a Microsoft windows application, communication between MS windows applications can be done through message passing.

7.0 CONCLUSION AND FUTURE WORK

The number of tools for building expert systems has increased significantly after the progress of expert systems in industry. Yet, there are few works about Arabic expert system shells. This paper presents KHABEER as one of pioneer projects of Arabic expert system tools. KHABEER is written in C language, which supports the goals of high portability, low cost, and ease of integration with external systems. KHABEER, supports Object Oriented Programming where it has 11 predefined classes and allows abstract and concrete class definitions and multiple inheritance.

Various features of slots are supported by KHABEER. KHABEER allows the declaration of message-handlers for defined classes. Manipulating instances of objects is supported through different functions in KHABEER. These functions include creating instances, re-initializing existing instances, reading slots, setting slots, deleting instances, instance query and other actions. These functions and other Object Oriented features are supported in ***Arabic*** by KHABEER version 2.

KHABEER provides a query system for determining, grouping and performing actions on sets of instances of user-defined classes that meet user-defined criteria. The user can use the query system to determine if a particular association set exists, he can save the set for future reference, or he can iterate an action over the set. KHABEER has Windows interface under running under MS. Windows. Researchers who are working in research areas, such as Arabic Language Understanding, Machine Translation, Semantic Representation of Arabic language and others, will find KHABEER as a good tool for them.

ACKNOWLEDGMENT

The Authors wish to acknowledge King Fahd University of Petroleum and Minerals (KFUPM) for utilizing the various facilities in preparation of this paper.

REFERENCES

[1] David W. Rolston, "Principles of Artificial Intelligence and Expert Systems Development", McGraw-Hill Book Company. 1988.

[2] William Mettrey, "A comparative Evaluation of Expert System tools," *IEEE Computer Magazine*, Vol 24, No. 2, pp19-31, Feb .1991.

[3] Chung S. Kim and Youngohc Yoon, "Selection of a good expert system shell for instructional purposes in business," *Information and Management* 23, pp 249-262, 1992.

[4] A. C. Stylianou, G. R. Madey and R. S. Smith, "Selection Criteria for Expert System Shells: A Socio-Technical Framework," *Communications of The ACM*, Vol. 35, No. 10, pp 30-48, October 1992.

[5] Mostafa M. Aref and Husni Al-muhtaseb, "KHABEER: (خبير) An Arabic Expert System Shell", *The 18th International Conference For Statistics, Computer Science, Scientific & Social Applications*, Cairo, Egypt, April, 1993.

[6] Husni A. Al-Muhtaseb and Mustafa M. Aref, "Arabic Technical Terms in Arabic Formal Languages", *Proceedings of The 3rd International Conference on Multi-lingual Computing,* University of Durham, UK, December 1992.

[7] Mostafa Aref and Husni Al-Muhtaseb, "KHABEER Reference Manual," A Technical Report, ICS Department, KFUPM, Dhahran, Saudi Arabia, (to be published)

[8] Husni A. Al-Muhtaseb, Mustafa M. Aref, and Ali Al-Kulaib, "Khool: KHABEER (خبير) Object Oriented Language", *Proceedings of the* 4th International Conference and Exhibition on Multi-lingual Computing, London, UK, April 1994.

[9] "CLIPS Reference Manual Version 4.3", Mission Planning and Analysis Division, Artificial Intelligence Section. NASA Johnson Space Flight Center, USA, 1989.

[10] "CLIPS Reference Manual Version 5.1", Software Technology Branch, *Lyndon B. Johnson Space Center*, USA, September 1991.

[11] George F. Luger, William A. Stubblefield, "Artificial Intelligence and the design of Expert System", The Benjamin/Cummings publishing Company، 1989.